HTML

Leereenheid 2

HTML

INTRODUCTIE

Het world wide web is in principe een verzameling documenten die op verschillende servers kunnen staan, en naar browsers verstuurd worden via het protocol HTTP. Voor HTML-documenten geldt dat ze links bevatten naar andere documenten, waardoor er 'gesurfd' kan worden. HTML-documenten vormen dus de basis van het web.

In deze cursus houden we zo streng mogelijk vast aan het principe van gescheiden verantwoordelijkheden. Het principe van gescheiden verantwoordelijkheden houdt in dat u HTML gebruikt waar het voor is bedoeld: om de betekenis aan te geven van wat er tussen de tags staat. Dat wordt aangeduid als 'semantische HTML'. Alles wat van belang is voor het uiterlijk van webpagina's, is de verantwoordelijkheid van CSS-regels (die in de volgende leereenheid aan bod komen). Alles wat van belang is voor de functionaliteit van de pagina is de verantwoordelijkheid van het JavaScript dat wordt meegestuurd; het grootste gedeelte van deze cursus heeft betrekking op JavaScript.

U krijgt hier geen compleet overzicht van HTML: het doel van deze cursus is niet om u op te leiden tot een webontwerper, maar om u technieken te leren die een webprogrammeur tot zijn of haar beschikking moet hebben. Daarbij hoort dat u HTML moet kunnen begrijpen en dat u gemakkelijk moet kunnen zoeken naar welke tags u kunt gebruiken en naar wat de betekenis van de tags is.

We behandelen in deze leereenheid twee voorbeelden van HTML-pagina's: een pagina die een artikel bevat met een kop, een footer, en twee kolommen waarvan de linkerkolom het artikel bevat (pas in de volgende leereenheid ziet u hoe u die kolommen naast elkaar kunt zetten), ingedeeld in secties, en een pagina met een uitgebreid formulier. We laten hier alleen de HTML-elementen zien die we nodig hebben om die pagina's op te bouwen.

De nadruk ligt op het begrijpen van de structuur van HTML.

LEERDOELEN

Na het bestuderen van deze leereenheid, wordt verwacht dat u:

- de functie van de doctype-declaratie kunt uitleggen en weet wat de betekenis is van de HTML5-doctype-declaratie
- kunt uitleggen wat metadata zijn en waarvoor ze gebruikt kunnen worden
- de syntax template van een basis HTML-pagina kunt geven
- kunt uitleggen wat de term semantische markup betekent, en wat het voordeel is die te gebruiken

- kunt aangeven wat semantische fouten en wat syntactische fouten in HTML zijn
- een pagina kunt schrijven in HTML met een aantal gegeven tags
- een element met attributen kunt gebruiken
- kunt uitleggen wat een property is en properties kunt gebruiken
- kunt aangeven wat een HTML validator doet, en er gebruik van kunt maken
- de controls van een formulier op de juiste wijze kunt gebruiken.

Studeeraanwijzingen
Het is niet de bedoeling dat u eigenschappen van de behandelde HTML-elementen uit uw hoofd gaat leren. Het gaat er om dat u met HTML leert werken, en de eigenschappen kunt vinden als u ze nodig hebt.

LEERKERN

1 HTML

1.1 HET DOCTYPE

Doctype

Elke HTML-pagina hoort te beginnen met een aanduiding van het *doctype*. De browser kan aan dat doctype zien hoe wat volgt geïnterpreteerd moet worden: het doctype geeft aan of het gaat om HTML of XHTML (een variant van HTML waarbij de regels voor de syntaxis van XML worden aangehouden), en om welke versie het gaat. Voor HTML5 ziet het doctype er als volgt uit:

```
<!DOCTYPE html>
```

Er kunnen zowel kleine letters als hoofdletters worden gebruikt, maar het is gebruikelijk om hoofdletters te gebruiken.

Deze aanduiding verschilt wezenlijk van de aanduiding bij vorige versies van HTML: bij vorige versies werd uitgebreide versie-informatie meegegeven. De reden dat de doctype-declaratie van HTML5 zoveel simpeler is dan van eerdere HTML-versies, is dat er een afspraak is dat HTML-bestanden die geschreven zijn volgens de HTML5-standaard ook zullen voldoen aan toekomstige standaarden van HTML. Er zullen dus geen versies van HTML verschijnen die maken dat bestaande HTML-bestanden moeten worden aangepast om met de tijd mee te kunnen. Toekomstige versies van HTML zullen wel nieuwe tags en attributen kennen (over tags en attributen leest u straks meer), maar er zal niets verdwijnen dat in de huidige standaard is vastgelegd.

Een browser die bovenstaande doctype-declaratie inleest, weet dat het gaat om een document in HTML5 of een hogere versie.

1.2 DE SYNTAXIS VAN HTML

In leereenheid 1 heeft u al de globale structuur van een HTML-document gezien. Een pagina met minimale hoeveelheid HTML, die voldoet aan de standaard, ziet er als volgt uit:

```
<!DOCTYPE html>
<html>
  <head>
    <title>De titel van de pagina</title>
  </head>
  <body>
  </body>
</html>
```

Element
Begintag
Eindtag

Na de doctype-declaratie ziet u een aantal *elementen* die bestaan uit een *begintag*, zoals `<head>`, en een *eindtag*, zoals `</head>`. Tussen begin- en eindtag kan tekst staan, zoals in het geval van het `title`-element, maar er kunnen ook andere elementen tussen staan, zoals in het geval van het `head`-element. De syntaxis van HTML schrijft voor dat de `nesting van elementen` klopt. Een constructie als:

```
<head><title>De titel</head></title>
```

is dus niet toegestaan.

Nesting van elementen

Het is goed gebruik om de nesting van elementen expliciet aan te geven door in te springen, maar het is niet voorgeschreven: de browser zal dus geen fout melden wanneer u dat niet zou doen.

Programmeeraanwijzing: Nesting van elementen
Maak de nesting van elementen in een HTML-bestand expliciet door in te springen bij een element dat tussen de begin- en eindtag van een ander element staat.

Inhoud

Bij sommige elementen mag de eindtag worden weggelaten. U zult bijvoorbeeld het `img`-element tegenkomen, waarmee u een afbeelding in een pagina kunt tonen. Dat element kent alleen een begintag, geen eindtag. Dat houdt in dat zo'n element geen *inhoud* kan hebben: de inhoud van een element is datgene (tekst of andere elementen) wat tussen begin- en eindtag staat. Het `img`-element is een voorbeeld van een *leeg element*. Het is gebruikelijk (maar niet vereist) om een leeg element te laten eindigen op `/>`.

Leeg element

Attributen

Elke begintag kan een aantal *attributen* meekrijgen. Een attribuut bestaat uit een naam en een waarde, en wordt geschreven als `naam="waarde"`. Een `img`-element kan er bijvoorbeeld als volgt uitzien:

```
<img src="wolk.png" alt="cumulus" width="90" height="90" />
```

Hierin zijn `src`, `alt`, `width` en `height` attributen, die een waarde krijgen die tussen aanhalingstekens staat. De waarde van een attribuut bestaat dus uit karakters tussen aanhalingstekens. Het is belangrijk om het onderscheid te maken tussen de *inhoud van een element* (in het geval van het `img`-element is er geen inhoud) en de *waarden van de attributen* van een element.

Inhoud van een element
Waarden van de attributen

1.3 SEMANTIEK

Semantiek

Semantische markup

Tags hebben als functie dat ze betekenis geven aan wat er tussen staat. Een ander woord voor betekenis is *semantiek*. U zult dan ook de term *semantische markup* kunnen tegenkomen. Die term houdt in dat tags in de eerste plaats bedoeld zijn om betekenis te geven aan hun inhoud, en uitdrukkelijk niet om een bepaald uiterlijk op de pagina te bereiken. Semantische markup houdt in: betekenisvolle markup.

De semantiek lijkt niet belangrijk. U kunt tenslotte aan het uiterlijk van elementen op een pagina wel zien of een stuk tekst bijvoorbeeld als kop fungeert of niet. Maar voor browsers voor blinden, die pagina's voorlezen, is de semantiek erg belangrijk. Datzelfde geldt voor zoekrobots, die aan een HTML-pagina belangrijke zoektermen voor zo'n pagina moeten ontlenen. Tags in HTML worden gebruikt door:
– browsers voor blinden, om aanwijzingen te hebben over hoe de tekst voor te lezen
– zoekrobots, om de status en betekenis van teksten te herkennen
– browsers, voor het vormgeven van de elementen (bij afwezigheid van een stylesheet).

Semantische markup is dus van belang voor browsers voor blinden en voor zoekrobots. Semantische markup zorgt er bovendien voor dat de verantwoordelijkheid voor het uiterlijk van een pagina exclusief bij CSS wordt gelegd, terwijl HTML de verantwoordelijkheid heeft voor de betekenis. Het draagt dus bij aan de scheiding van verantwoordelijkheden.

HTML5 is wat semantiek betreft een sprong voorwaarts ten opzichte van eerdere versies. Er zijn elementen bijgekomen die aangeven hoe de indeling van een pagina is (met header, footer en sections bijvoorbeeld), en er zijn allerlei extra attributen voor `input`-elementen die iets vertellen over de soort input die wordt verwacht.

1.4 VALIDATIE

Browsers zijn niet strikt ten opzichte van fouten in de syntaxis van HTML: als een browser 'begrijpt' wat bedoeld wordt, bouwt de browser op basis daarvan een DOM-boom op. Dat is anders wanneer er XHTML5 in plaats van HTML5 wordt gehanteerd: dan straft de browser elk foutje af door alleen een foutmelding te laten zien.

Het voordeel van het toestaan van fouten in HTML is dat gebruikers niet direct een lege pagina te zien krijgen wanneer er bijvoorbeeld een eindtag is vergeten bij een element waarbij een eindtag verplicht is; het nadeel is dat je als ontwikkelaar niet zeker weet of je geen fout in de HTML hebt: het feit dat de browser laat zien wat je verwacht is geen garantie dat de HTML foutvrij is.

Weblink: HTML validator

Het is daarom een goede gewoonte om regelmatig de HTML op fouten te controleren met behulp van een validator. Het World Wide Web consortium heeft er één (zie weblink).

Syntaxis

Een validator controleert de *syntaxis*: de 'grammatica' van de code. Syntaxisfouten zijn:
– ontbrekende verplichte elementen
– ontbrekende verplichte attributen
– verkeerde nesting van tags
– elementen die niet in de specificatie voorkomen
– attributen die niet in de specificatie voor een element staan
– attributen met waarden die niet zijn toegestaan
– ontbrekende tags.

Een validator kan niet zien of er semantische fouten zijn gemaakt. Wanneer u een `p`-element gebruikt voor een heading, of een onderschrift bij een afbeelding in een `table`-element stopt, zijn dat *semantische* fouten, die de validator er niet uithaalt. Die moet u zelf in de gaten houden.

OPGAVE 2.1

Ga naar de validator, en kies daar voor 'Validate by Direct Input'. Vul de minimale webpagina uit paragraaf 1.2 daar in (u vindt de pagina in de bouwsteen bij deze leereenheid onder `pagina/minimaal.html`), en kies 'Validate'. Bekijk de meldingen.

Haal vervolgens het gehele `title`-element weg en kies 'Revalidate'.

2 Tags: betekenis en structuur

Weblink: W3C specificatie
Weblink: periodieke tabel

In de specificatie van HTML (zie weblink) zijn de verschillende elementen in groepen ingedeeld. In de weblink 'Periodieke tabel' ziet u die groepen visueel terug. We zullen hier een beperkt aantal elementen bespreken, ingedeeld in diezelfde groepen.

Bij het bespreken van de elementen werken we een voorbeeld uit: een pagina die als onderwerp de elementen van de `head` zal hebben. Die pagina zullen we in de volgende leereenheid gaan vormgeven.

2.1 ELEMENTEN VOOR DE HEAD

Weblink: Meer over HTML5

Wanneer u meer over de besproken elementen wilt weten, of wilt bekijken welke elementen we hebben overgeslagen, kunt u de sites bekijken die we in de weblink hebben opgenomen.

De `head` van een HTML-pagina bevat metadata, die kunnen worden aangegeven middels de elementen uit tabel 2.1.

TABEL 2.1 Elementen van de head

tag	*betekenis*	*verplicht*	*inhoud*	*eindtag*
`head`	begrenzing head	ja	elementen	ja
`title`	titel van het document	ja	tekst	ja
`meta`	metadata	nee	geen	nee
`link`	link naar ander document	nee	geen	nee
`script`	JavaScript	nee	mag (tekst)	ja

head-element
title-element

Van al deze elementen zijn alleen het *head-element* en het *title-element* verplicht (het omhullende `html`-element is ook verplicht). De structuur van het `title`-element is simpel: een begintag, een eindtag en de tekst van de titel daartussen. Ook de structuur van de `head` is simpel: een begin- en een eindtag en daartussen de elementen van de `head`. Voor de andere elementen ligt dat niet zo eenvoudig: die hebben attributen nodig.

meta-element
name-attribuut
content-attribuut

Het *meta-element*, voor metadata (gegevens over het document), kent een *name-attribuut* en een *content-attribuut*. Met de `name` wordt het type metadata aangegeven, zoals `description` of `author`. Het `content`-attribuut geeft dan de uiteindelijke beschrijving of naam van de auteur.

Weblink: Metadata

De HTML5-specificatie geeft een opsomming van waarden die gebruikt mogen worden voor het `name`-attribuut van het `meta`-element (zie weblink).

charset-attribuut

Het `meta`-element kent ook een *charset-attribuut*, waarmee u de character encoding van uw document kunt aangeven.

link-element
rel-attribuut

Het *link-element* kent een *rel-attribuut,* dat staat voor 'relation'. Veelgebruikte relaties zijn `next` en `prev` voor de documenten die logisch volgen op of voorafgaan aan het huidige document. Een nog bekender gebruik van het `rel`-attribuut is `stylesheet`, om een bij de pagina behorende stylesheet in CSS mee aan te geven. De documenten worden meegegeven via een *href-attribuut*.

href-attribuut

Weblink: Link types

De specificatie laat zien welke opsomming van waarden van het `rel`-attribuut van het `link`-element gebruikt mogen worden (zie weblink).

script-element
src-attribuut

Bij het *script-element* gebeurt iets soortgelijks, alleen wordt hier het script aangegeven door middel van het *src-attribuut*. Het `script`-element kent in tegenstelling tot het `meta`- en het `link`-element wel een eindtag: het element kan ook gebruikt worden zonder attribuut; in dat geval wordt de JavaScript-code als inhoud meegegeven tussen de begin- en de eindtag.

Wanneer u daar gebruik van zou maken, staat de JavaScript-code in de HTML-code. Dat is strijdig met het uitgangspunt dat we JavaScript en HTML gescheiden willen houden. Ook wanneer u geen code tussen begin- en eindtag zet maar gebruikmaakt van een verwijzing naar een bestand via het `src`-attribuut, moet u het `script`-element een eindtag geven. Er staat dan niets tussen begin- en eindtag.

Programmeeraanwijzing: Script met src
Verwijs in de HTML altijd met een `src`-element naar een script, en neem niet de JavaScript-code op in de HTML.

Een voorbeeld van een uitgebreide `head` is:

```
<head>
  <title>
    De tags binnen de head
  </title>
  <meta charset="UTF-8" />
  <meta name="description" content="We beschrijven de tags
die gebruikt kunnen worden binnen de head, met hun
mogelijkheden en bijzonderheden" />
  <meta name="author" content="Sylvia Stuurman en Harrie
Passier" />
  <meta name="keywords" content="head, title, meta, link,
style, script" />
  <link rel="next" href="secties.html" />
  <link rel="stylesheet" href="stijl.css" />
  <script
src="http://code.jquery.com/jquery.min.js"></script>
  <script src="gedrag.js"></script>
</head>
```

OPGAVE 2.2

Neem in de head een `link`-element op dat aangeeft wat de pagina is die logsich gezien aan deze voorafgaat (u bent uiteraard geheel vrij daar zelf een bestandsnaam voor te verzinnen). U vindt een pagina met de uitgangscode in de bouwsteen bij deze leereenheid onder `pagina/head.html`. Controleer met behulp van de validator of uw oplossing syntactisch gezien aan de standaard voldoet.

2.2 ELEMENTEN VOOR SECTIES

Buiten de elementen in de `head` die verplicht zijn, en het `html`-element, zijn er geen elementen die verplicht aanwezig moeten zijn op een pagina.

TABEL 2.2 Tags voor decties

tag	*betekenis*	*inhoud*	*eindtag*
body	begrenzing body	elementen	ja
article	artikel	elementen	ja
section	sectie	elementen	ja
h1 t/m h6	headings	tekst	ja
header	header	elementen	ja
footer	footer	elementen of tekst	ja
address	contactinformatie	elementen of tekst	ja
aside	hoort er zijdelings bij	elementen of tekst	ja
nav	navigatie	elementen	ja

In tabel 2.2 ziet u een selectie van de elementen die te maken hebben met secties. Elementen die hieronder vallen hebben vooral te maken met de structuur van de pagina. Een pagina zal vaak één of meer *article-elementen* bevatten. Een `article` is een onderdeel dat in principe losstaat van andere onderdelen, zoals een krantenartikel of een blog-entry op een pagina waar de laatste entries onder elkaar staan.

article-element

section-element

Een `article`-element kan in meerdere *section-elementen* zijn ingedeeld. Een `section` is te vergelijken met een hoofdstuk: het is een deel van een `article` dat onderscheiden kan worden van de rest van het `article`, of van andere `sections`.

header-element
h1-element

De gehele pagina, elk artikel en elke sectie kan een *header-element* bovenaan hebben, met één of meer headings: een *h1-element* tot en met een `h6`-element (de headings kunnen ook los gebruikt worden). Het is uiteraard de bedoeling dat de headings netjes in volgorde worden gebruikt: een `h2` onder een `h1` enzovoort.

footer-element
address-element

Een pagina, artikel en zelfs een sectie kan onderaan een *footer-element* bevatten, waarin vaak contactinformatie te vinden is in een *address-element*.

aside-element

nav-element

Ten slotte is het mogelijk om zijdelingse informatie (dat is vaak een tekst in de marge, maar het kan ook worden vormgegeven als blokjes tekst in een artikel) in een *aside-element* te stoppen, en navigatie zoals tabs, een menu of broodkruimels, horen thuis in een *nav-element*.

De structuur van de pagina die we hier opbouwen kan er als volgt uitzien:

```
<body>
  <article>
    <header>
      <nav class="tabs">...</nav>
      <h1>De tags van de head</h1>
      <nav class="breadcrumbs">...</nav>
    </header>
    <section class="introduction">
      <h2>Introductie</h2>
      <aside>De specificatie vindt u bij het W3C</aside>
            ...
    </section>
    <section>
      <h2>De head tag</h2>
          ...
    </section>
    <section>
      <h2>De title tag</h2>
          ...
    </section>
  </article>
  <aside class="sidebar">
    <section>
      <nav role="menu">
        <h2>Menu</h2>...
      </nav>
    </section>
    <section>
      <h2>Links</h2>
    </section>
    <section>
      <h2>Zoeken</h2>
    </section>
  </aside>
  <footer>
    <address>Open Universiteit Nederland</address>
  </footer>
</body>
```

class-attribuut
role-attribuut

U ziet drie verschillende `nav`-elementen (die we nog geen inhoud hebben gegeven). Om een idee te hebben van de functie van die `nav`-elementen, gebruiken we attributen die aan elk element mogen worden meegegeven: tweemaal het *class-attribuut* en eenmaal het *role-attribuut*. Er bestaat geen standaard voor de waarden van dat `class`-attribuut, maar zoekrobots herkennen veelgebruikte namen (zoals `breadcrumbs`) wel. We hebben daarom Engelse namen gebruikt voor de waarden van het `class`-attribuut. Het `class`-attribuut kunt u dus zien als een middel om – vooral voor uzelf maar soms ook voor zoekrobots – nog meer betekenis te geven aan tags.

Wanneer een bepaalde waarde voor het `class`-attribuut erg vaak wordt gebruikt, is er een kans dat er in een volgende HTML-versie een nieuw element voor wordt gespecificeerd.

Weblink: Role attribuut

Behalve het `class`-attribuut bestaat er ook een `role`-attribuut, dat aan elk element kan worden meegegeven. Waarden voor het `class`-attribuut kunt u zelf bepalen; mogelijke waarden voor het `role`-attribuut zijn gespecificeerd (zie weblink). De meeste van die waarden hebben geen functie meer, omdat er nu speciale HTML-elementen voor zijn. In eerdere versies van HTML kon er aan elementen bijvoorbeeld als `role navigation` worden meegegeven; nu is daar het `nav`-element voor. De `role menu` is wel zinvol: daar bestaat geen HTML-element voor.

Om het `aside`-element voor de sidebar (met het menu, een verzameling links en een zoekbox) te kunnen onderscheiden van de `aside`-elementen in het artikel zelf, krijgt ook dat element een `class`-attribuut.

Programmeeraanwijzing: Elementen, class en role
Kijk eerst of er een HTML-element bestaat met de betekenis die u zoekt. Indien dat niet het geval is, kijk of er een role is gedefinieerd. Indien ook dat niet het geval is, definieer dan een class.

OPGAVE 2.3
Schrijf een sectie (de inhoud kunt u met puntjes aangeven, zoals in het codevoorbeeld) met als titel 'De meta tag', met een subsectie 'Namen van Metadata'. U vindt een pagina met de uitgangscode in de bouwsteen bij deze leereenheid onder `pagina/secties.html`. Welke van de `h1-h6`-elementen zou u gebruiken voor de titel van die subsectie?

2.3 TAGS VOOR HET GROEPEREN VAN CONTENT

TABEL 2.3 Tags voor het groeperen van content

tag	*betekenis*	*inhoud*	*eindtag*
`p`	alinea	tekst	ja
`hr`	horizontale lijn	geen	nee
`pre`	preformatted	tekst	ja
`blockquote`	citaat	tekst	ja
`ol`	ordered list	`li`	ja
`ul`	unordered list	`li`	ja
`li`	item in lijst	tekst	ja
`dl`	description list	`dl`, `dt`	ja
`dt`	term	tekst	ja
`dd`	beschrijving	tekst	ja
`figure`	figuur	elementen	ja
`figcaption`	titel van figuur	tekst	ja
`div`	geen betekenis	elementen	ja

p-element

In tabel 2.3 ziet u een selectie van de elementen die te maken hebben met het groeperen van content. De meest gebruikte van deze elementen is het *p-element*, om een alinea (Engels: paragraph) te markeren: de tekst binnen de begin- en eindtag is een alinea.

ol-element
ul-element
li-element
dl-element
dt-element
dd-element

Andere veelgebruikte elementen zijn lijsten. Geordende lijsten (waarvan de browser de items nummert) kunt u maken met het *ol-element*; ongeordende (waarvan de browser de items weergeeft met een bolletje) kunt u creëren met het *ul-element*. De items maakt u in beide gevallen met *li-element*en. Een derde vorm van een lijst is de description list, met het *dl-element*. In plaats van `li`-elementen bevat die lijst paren van een *dt-element* met een term, en een *dd-element* met de beschrijving.

pre-element

Het *pre-element* gebruikt u om tekst inclusief tabs en regeleinden weer te geven: wat tussen `pre`-tags staat, wordt precies zo weergegeven als het in de HTML staat.

figure-element
figcaption-element
hr-element
blockquote-element

Er is een tag voor figuren (het *figure-element*) die bijvoorbeeld een afbeelding of een tabel kunnen bevatten. U kunt de figuur een titel geven met het *figcaption-element* dat binnen het `figure`-element staat. Verder zijn er elementen voor een horizontale lijn (het *hr-element*) en voor een citaat uit een andere bron (het *blockquote-element*).

Alle elementen hebben een begin- en een eindtag, behalve de horizontale lijn: die maakt u simpelweg met `<hr />`.

div-element

Ten slotte is er een element dat bedoeld is om elementen te groeperen zonder dat er een speciale betekenis aan wordt gehecht: het *div-element*. U kunt dat element gebruiken om aan te geven dat groepen elementen bij elkaar horen. Verderop in deze leereenheid zult u daar een voorbeeld van zien.

Met deze tags is het mogelijk om bijvoorbeeld de `nav` die de broodkruimels voorstelt aan te vullen:

```
<nav class="breadcrumbs">
  <ul>
    <li class="crumb">Home</li>
    <li class="crumb">HTML</li>
    <li class="crumb">Tags</li>
    <li class="crumb active">Head</li>
  </ul>
</nav>
```

Meerdere classes

Hier geven we bij elk item aan dat het om een broodkruimel gaat. Bij het laatste item ziet u dat er twee classes zijn toegewezen: `crumb` en `active`. *Meerdere classes* toewijzen aan een element is dus mogelijk door ze met een spatie van elkaar te scheiden en gezamenlijk tussen aanhalingstekens te zetten.

Ook in dit geval is het weer zo dat u met het toevoegen van classes meer betekenis aan de elementen kunt geven dan alleen met tags mogelijk is. Dat de betekenis van die classes niet is vastgelegd in een specificatie, doet er niet toe: het is duidelijk voor de lezer van de code (en u kunt de classes later gebruiken vanuit CSS).

Verder kan de tekst in paragrafen worden geschreven:

```
<section>
  <h2>De title tag</h2>
  <p>De title-tag is verplicht aanwezig: wanneer u een HTML-
pagina zonder title-tag valideert, zult u een foutmelding te
zien krijgen.</p>
  <p>De structuur van deze tag is simpel: een begintag, een
eindtag, en de titel komt daartussen.</p>
```

OPGAVE 2.4

Vul het `nav`-element met class `tabs` ook met een ongeordende lijst. Als items kunt u nemen: HTML, CSS en JavaScript. Geef de items class `tab` mee, en geef de relevante tab ook class `active` mee.

2.4 TAGS VOOR TEKST

TABEL 2.4 Tags voor tekst

tag	*betekenis*	*inhoud*	*eindtag*
`a`	hyperlink	tekst	ja
`strong`	sterk	tekst	ja
`em`	emphasis, nadruk	tekst	ja
`mark`	gemarkeerd	tekst	ja
`code`	computercode	tekst	ja
`span`	geen betekenis	tekst	ja

In tabel 2.4 ziet u een selectie van de elementen waarmee u een stuk tekst van extra betekenis kunt voorzien. Deze elementen kunnen overal worden gebruikt waar als inhoud Tekst is aangegeven. Ze kunnen tussen andere tekst in staan: deze elementen gedragen zich als tekst. Bij het `p`-element bijvoorbeeld zag u dat het tekst kan bevatten; de elementen die u hier ziet, kunnen gebruikt worden om een of meerdere woorden in zo'n tekst extra betekenis te geven.

strong-element
em-element
mark-element
code-element

Zo kunt u een woord of meerdere woorden markeren als 'sterk' door het woord of de woorden tussen begin- en eindtag van het *strong-element* te zetten. Nadruk geeft u aan met het *em-element*. Woorden als het ware met een markeerstift markeren gaat met het *mark-element* (waarbij is aangetekend dat niet alle browsers daar zonder CSS iets van laten zien). En om van een stuk tekst aan te geven dat het gaat om computercode, kunt u het *code-element* gebruiken.

span-element

Zoals er een element zonder betekenis is dat groepen elementen bij elkaar kan houden, is er ook een element zonder betekenis dat een stukje tekst van de rest van de tekst kan scheiden: het *span-element*.

We kunnen de tekst over de `title`-tag nu als volgt aanpassen:

```
<p>Een voorbeeld van het gebruik van deze tag is:
  <code>
    &lt;title&gt;De titel van een pagina&lt;/title&gt;
  </code>
</p>
```

Merk hier de merkwaardige tekenreeksen op die staan op de plaatsen waar u < en > zou verwachten. Die tekens kunt u in HTML niet gebruiken, omdat de browser ervan uit zou gaan dat u daar een tag mee wilt aangeven. Om die tekens letterlijk te gebruiken moet u gebruikmaken van de *speciale tekens* van HTML. Zo'n teken begint met een ampersand (&) en eindigt met een puntkomma. De letters `lt` en `gt` staan voor lower than en greater than. De weblink laat meer speciale karakters voor HTML zien.

Speciale tekens

Weblink: Special characters

OPGAVE 2.5

Markeer in de tekst over de `title`-tag het eerste woord title-tag als sterk, en geef de woorden begintag en eindtag nadruk.

De belangrijkste van deze elementen is het *a-element*, voor hyperlinks. Tussen de begin- en eindtag van dit element staat de tekst waarop geklikt kan worden. De URL waar dan naar toe gesprongen wordt, staat in een *href-attribuut*. Een `a`-element kan ook nog een *title-attribuut* krijgen, waarvan de tekst als een soort tooltip verschijnt wanneer de gebruiker de muis boven de link houdt.

a-element

href-attribuut

title-attribuut

We kunnen de tekst in het `aside`-element nu als volgt aanpassen:

```
<aside>De
  <a href="http://www.w3.org/TR/html5/"
     title="HTML5 specificatie">specificatie
  </a> vindt u bij het W3C
</aside>
```

2.5 TABELLEN

TABEL 2.5 Tags voor tabellen

tag	*betekenis*	*inhoud*	*eindtag*
`table`	tabel	`thead, tbody, tr, caption`	ja
`thead`	heading van de tabel	`tr`	ja
`tbody`	het deel met de data	`tr`	ja
`tr`	rij	`th` of `td`	ja
`th`	cel in de heading	tekst	ja
`td`	cel in de body	tekst	ja
`caption`	titel van de tabel	tekst	ja

In tabel 2.5 ziet u een selectie van de elementen waarmee u een tabel kunt opstellen. Het *table-element* geeft aan dat wat tussen begin- en eindtag staat een tabel is. Een tabel kan worden ingedeeld in een heading, met een *thead-element*, en een body, met een *tbody-element*. Het is niet verplicht die te gebruiken (maar het voegt extra betekenis toe). Een rij in een tabel wordt aangegeven met een *tr-element*: die kunt u zowel in de head als in de body van de tabel gebruiken. Een cel in een rij wordt aangegeven met een *th-element* als het om een kopje gaat, en met een *td-element* als het om een cel met gegevens gaat.

table-element

thead-element

tbody-element

tr-element

th-element

td-element

Met deze kennis kunnen we in de introductie van onze pagina het volgende zetten:

```
<p>Op deze pagina behandelen we een aantal elementen van de
head. Hier alvast een referentie:</p>
<table>
  <caption>De elementen van de head</caption>
  <thead>
    <tr>
      <th>Element</th>
      <th>Beschrijving</th>
      <th>Verplicht</th>
      <th>Begin- en eindtag</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>head</td>
      <td>Begrenzing head</td>
      <td>V</td>
      <td>V</td>
    </tr>
    <tr>
      <td>title</td>
      <td>Titel</td>
      <td>V</td>
      <td>V</td>
    </tr>
    <tr>
      <td>meta</td>
      <td>Metadata</td>
      <td>-</td>
      <td>-</td>
    </tr>
    <tr>
      <td>link</td>
      <td>Link naar andere documenten</td>
      <td>-</td>
      <td>-</td>
    </tr>
    <tr>
      <td>script</td>
      <td>JavaScript</td>
      <td>-</td>
      <td>V</td>
    </tr>
  </tbody>
</table>
```

OPGAVE 2.6

a Voeg aan de sectie Namen van Metadata die u in een van de vorige opdrachten hebt gemaakt, een tabel toe met een naam en een beschrijving. Als namen voegt u toe: author, description en keywords; als beschrijving: auteur, beschrijving en steekwoorden.

b U bent eerder in deze leereenheid een element tegengekomen dat u als alternatief voor een tabel zou kunnen gebruiken in dit geval. Kunt u argumenten voor dat andere element bedenken, en argumenten voor een tabel?

TABEL 2.6 Tags voor ingebedde content

tag	*betekenis*	*inhoud*	*eindtag*
`img`	afbeelding	geen	nee
`object`	een bron van elders	elementen	ja
`canvas`	tekenen met JavaScript	elementen	ja
`video`	video	elementen	ja
`iframe`	een andere webpagina	elementen	ja

canvas-element

In tabel 2.6 ziet u een selectie van de elementen waarmee u afbeeldingen, video's of documenten kunt inbedden in een pagina. Het *canvas-element* is hier een vreemde eend in de bijt: er wordt niets mee ingebed, maar het biedt de JavaScript-programmeur de mogelijkheid om het dynamisch te vullen, op eenzelfde manier als mogelijk is met Flash. Onderaan de pagina ziet u links naar een aantal voorbeelden waaraan u kunt zien wat er mogelijk is. We zullen het `canvas`-element niet bespreken in deze cursus, maar u krijgt voldoende kennis mee om er met behulp van een tutorial zelf mee te kunnen werken.

Weblink: canvas-element

width-attribuut
height-attribuut

Alle elementen uit deze groep kunnen een *width-attribuut* en een *height-attribuut* meekrijgen. Het is bijzonder aan te bevelen om dat te doen: de browser weet dan hoeveel ruimte er gereserveerd kan worden, en kan verder met de opbouw van de rest van de pagina voor de externe bron is binnengehaald. Wanneer dit wordt overgelaten aan CSS, moet de browser alles herberekenen zodra de CSS verwerkt is. De waarde van het attribuut is een getal, dat het aantal pixels voorstelt. U geeft dus geen eenheid mee aan het getal.

object-element
data-attribuut
type-attribuut
Fallback content

In principe is het *object-element* voldoende om externe bronnen mee te embedden: daar is het voor bedoeld. Het `object`-element kent een *data-attribuut* waarin de URL van de externe content kan worden meegegeven, een *type-attribuut* waarin staat aangegeven wat het type van de content is, en afhankelijk van de soort content zijn er meer attributen. Tussen begin- en eindtag van het `object`-element staat *fallback content*: content (tekst of een element) die wordt weergegeven door de browser wanneer de browser niet in staat is de door het attribuut `data` aangegeven bron weer te geven. Als het element dat als fallback wordt gebruikt zelf ook een fallback content heeft, wordt de fallback content dus genest.

Ook het `canvas`- en het `video`-element kennen een dergelijke fallback content.

img-element
alt-attribuut
src-attribuut

Het *img-element* is speciaal voor afbeeldingen, en kent het verplichte *alt-attribuut*, waarmee een tekst wordt meegegeven die de browser kan weergeven wanneer er geen afbeeldingen geladen kunnen worden (zoals in een browser voor blinden, of voor zoekrobots), en de URL van de afbeelding komt terecht in het *src-attribuut*. Een `img`-element heeft dus in plaats van fallback content tekst in het `alt`-attribuut. Dat heeft een soortgelijke functie, maar heeft minder mogelijkheden (omdat er geen elementen gebruikt kunnen worden).

video-element Het *video-element* is voor video (bijvoorbeeld bestanden met extensie `.mpg` of `.ogg`), en kent eenzelfde soort fallback content als het `object`-element.

iframe-element Het *iframe-element* kunt u gebruiken om een andere pagina mee in te bedden. Het wordt bijvoorbeeld gebruikt wanneer u een YouTube-filmpje wilt inbedden: u bedt dan een pagina van YouTube in waarop de (tamelijk ingewikkelde) code met een `object`-element staat.

Met deze kennis kunt u nu een filmpje invoegen in de sectie over het `meta`-element:

```
<section>
  <h2>De meta tag</h2>
  <figure>
    <figcaption>Online les over Meta tags</figcaption>
    <iframe width="560"
            height="315"
            src="http://www.youtube.com/embed/Rq6cxnaPEFI"
            frameborder="0"
            allowfullscreen>
    </iframe>
  </figure>
  ...
</section>
```

Ook kunt u een afbeelding toevoegen aan de introductie:

```
<section class="introduction">
  <h2>Introductie</h2>
  <figure>
    <figcaption>Zo ziet de head van een pagina er uit
    </figcaption>
    <img src="html.jpg"
         width="400"
         height="300"
         alt="code voor de head van een HTML-pagina" />
  </figure>
  ...
</section>
```

OPGAVE 2.7

Voeg aan de sectie over het `meta`-element twee afbeeldingen toe: `title-tags.png` en `meta-description.png`.
Voeg tekst toe waarin u uitlegt dat zoekmachines het `title`-element en het `meta`-element met als name `description` gebruiken om hun zoek-resultaten op te stellen.

3 Formulieren

De communicatie tussen browser en webserver die we tot nu toe hebben gezien, bestond alleen uit het opvragen van een nieuwe HTML-pagina. De inhoud van zo'n HTML-pagina is statisch: elke keer dat we de pagina opvragen is de inhoud hetzelfde.

Deze manier van communiceren is niet toereikend om bijvoorbeeld een offerte voor een autoverzekering aan te vragen. De inhoud van de offerte, bijvoorbeeld de premie per maand, is afhankelijk van één of meer variabelen, zoals het merk en type auto en de soort verzekering.

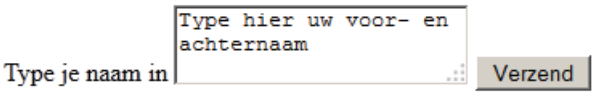
In deze paragraaf over formulieren leert u dat u data naar een webserver kunt sturen. De webserver kan van alles doen met deze data. Een achterliggende applicatie kan deze meegestuurde data bijvoorbeeld gebruiken om andere data uit een database te selecteren, waarna een backoffice-syteem aan het werk wordt gezet om een offerte op te stellen en in de vorm van een HTML-pagina terug te sturen.

Tabel 2.7 laat een aantal elementen zien die bij formulieren horen.

TABEL 2.7 Tags voor formulieren

tag	*betekenis*	*inhoud*	*eindtag*
`form`	formulier	elementen	ja
`fieldset`	samenhangend deel van formulier	elementen	ja
`legend`	titel van fieldset	tekst	ja
`label`	tekst bij invoerveld	tekst	ja
`input`	invoerveld	geen	nee
`select`	drop-down lijst	`option` en `optgroup`	ja
`optgroup`	deel lijst	`option`	ja
`option`	keuze	tekst	ja
`textarea`	tekstveld	tekst	ja

Figuur 2.1 toont een eenvoudig formulier in een webbrowser.



FIGUUR 2.1 Een eenvoudig formulier

Een gebruiker voert een voor- en achternaam (bijvoorbeeld Harrie Passier) in het tekstveld in en drukt daarna op de knop 'Verzend'. Vervolgens wordt de ingevulde voor- en achternaam verstuurd naar de server, waar een nieuwe HTML-pagina wordt samengesteld met, in dit geval, de tekst 'Hallo Harrie Passier!'.

Essentie formulier

Met een formulier kunnen we een programma op een server aanroepen en daarbij de nodige parameters in de vorm van *naam-waardeparen* meesturen.

naam-waardeparen

In ons voorbeeld is dat maar één parameter, namelijk het naam-waardepaar `'naam = Harrie Passier'`. Het woord '`naam`' is de waarde van het *name-attribuut* van het `form`-element.

name-attribuut

Het serverprogramma leest deze parameter uit en kan aan de hand hiervan een nieuwe pagina samenstellen en als antwoord terugsturen.

Let op!

Wanneer er meerdere parameters zijn, kan het serverprogramma aan de hand van de parameternaam de juiste parameter selecteren. Het is daarom noodzakelijk dat de namen van parameters binnen een formulier *uniek* zijn, en dat houdt in dat de waarden van de `name`-attributen uniek moeten zijn.

De bijbehorende HTML-code ziet er als volgt uit:

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <title>Hallo Service</title>
5.   </head>
6.   <body>
7.     <form method="get"
action="http://localhost:8080/WAC_le02_HalloService/Hallo">
8.       <label>Type je naam in</label>
9.       <textarea name="naam">
10.         Type hier uw voor- en achternaam
11.       </textarea>
12.       <input type="submit" value="Verzend" />
13.     </form>
14.   </body>
15.  </html>
```

De HTML bevat een `form`-element (regel 7 en 13) dat het formulier representeert. Het `form`-element bevat een `label`-, een `textarea`- en een `input`-element. Het `input`-element is van het type `submit` en representeert de verzendknop.

form-element

Een *form-element* groepeert een aantal user interface-elementen en maakt het mogelijk om gegevens naar een webserver te sturen. Deze user interface-elementen worden ook wel *controls* genoemd. Voorbeelden hiervan zijn labels, tekstvelden en knoppen. Controls gedragen zich als tekst en worden dus naast elkaar geplaatst.

Controls

Het `form`-element heeft een begin- en een eindtag, heeft als inhoud form controls, en kan twee attributen bevatten: het `method`- en het `action`-attribuut. Het *method-attribuut* kan twee waarden hebben, GET of POST. Op het verschil tussen beide komen we in de leereenheid over de communicatie tussen client en server terug.

method-attribuut

Het *action-attribuut* specificeert de URL van het programma op de webserver dat wordt aangeroepen. Het `form`-element zorgt ervoor dat, nadat op de submit-knop is geklikt, de gegevens van het formulier worden meegestuurd naar het programma op de server. Welke gegevens dat zijn, wordt bepaald door de controls die het `form`-element bevat. In ons geval gaat het alleen om een `textarea`-element.

action-attribuut

Het *textarea-element* kan meerdere regels tekst bevatten. Het `textarea`-element heeft een begin- en eindtag en kan als inhoud tekst bevatten (de gebruiker kan die tekst dan veranderen), maar de inhoud kan ook ontbreken (de gebruiker kan dan tekst invoeren). Het `textarea`-element kan optioneel de attributen *rows* en *cols* bevatten: ze definiëren het aantal rijen en kolommen dat in het tekstveld aanwezig is. Als er meer tekst wordt ingevoerd dan er ruimte is, verschijnt er automatisch een scrollbar.

textarea-element

rows-attribuut

cols-attribuut

Om het aantal karakters dat kan worden ingevoerd te begrenzen, kan het *maxlength-attribuut* worden gebruikt.

maxlength-attribuut

Wanneer een formulier met een `textarea` naar een server wordt verstuurd, wordt het bijbehorende naam-waardepaar als volgt samengesteld:

name-attribuut – De naam van de parameter is gelijk aan de waarde van het *name-attribuut*.

value – De waarde van de parameter is gelijk aan de *value* van het `textarea`-element, die bij dit element gelijk is aan de actuele inhoud van het element.

Het resultaat in ons voorbeeld is dus: '`naam = Harrie Passier`'.

submit button *type-attribuut* Het volgende element in het voorbeeldformulier is een *submit button*. Deze wordt gespecificeerd door het *type-attribuut* in het `input`-element de waarde `submit` te geven. De algemene syntaxis is als volgt:

```
<input type="submit" value="verzend" [overige-attributen] />
```

value-attribuut Het `input`-element is een leeg element. De waarde van een eventueel *value-attribuut* is de tekst die op de knop wordt afgebeeld. Na aanklikken worden de diverse naam-waardeparen in het formulier verzameld en naar de URL verstuurd die staat aangegeven in het `form`-element.

3.2 OVERIGE FORM-ELEMENTEN

input-element De submit button is een vorm van het *input-element*. Preciezer geformuleerd: het `input`-element *representeert* een *getypeerd* dataveld waarmee een gebruiker data kan invoeren. 'Representeren' betekent hier dat elk `input`-element door de waarde van het type een bepaalde semantiek heeft (bijvoorbeeld submit knop of tekstveld). 'Typeren' betekent hier dat we een waarde voor het type van het `input`-element meegeven. Dit betekent dat we met het `input`-element een reeks getypeerde `input`-elementen kunnen specificeren.

Tabel 2.8 toont de attributen van het `input`-element, waaronder het `type`-attribuut, en enkele waarden die deze kunnen aannemen.

TABEL 2.8 Attributen voor het input-element

attribuut	*waarden*	*beschrijving*
`type`	`button, checkbox, color, date, email, file, hidden number, password, radio, submit, tekst, …`	type control
`name`	tekst	unieke naam
`value`	tekst	de waarde van de control
`placeholder`	tekst	laat een voorbeeld zien
`required`	geen waarde	al dan niet aanwezig

– Het type *button* laat een knop zien; de waarde van een eventueel *value*-attribuut is de tekst op de knop.
– Het type *checkbox* wordt verderop besproken.
– De typen *color* en *date* komen in de leereenheid over formvalidatie aan bod.

– Het type *email* zorgt ervoor dat er een foutmelding ontstaat wanneer wat is ingevuld geen e-mailadres is.
– Het type *file* geeft een bladerfunctie waarmee een bestand op de harde schijf van de gebruiker kan worden geselecteerd.
– Het type *hidden* zorgt ervoor dat de gebruiker de control niet ziet (de waarde ervan wordt wel naar de server gestuurd, en soms kan het handig zijn over zo'n control te beschikken).
– Het type *number* wordt in de leereenheid over formvalidatie behandeld.
– Het type *password* zorgt ervoor dat de waarde die wordt ingevuld in de vorm van sterretjes wordt getoond.
– Het type *radio* wordt verderop behandeld.
– Het type *submit* is eerder besproken.
– Het type *text* wordt verderop besproken.

Er zijn meer typen dan we hier hebben aangegeven.

name-attribuut

Het *name-attribuut* bevat de naam van de parameter die naar de server-applicatie wordt gestuurd. Onder deze naam kan het programma op de server de waarde van de parameter vinden.

value-attribuut

Het *value-attribuut* specificeert de waarde van het `input`-element. De gebruiker kan deze waarde in de meeste gevallen invullen en veranderen. Het is deze waarde die in combinatie met de naam naar de server wordt gestuurd. Bij een knop wordt de waarde van dit attribuut afgebeeld op de knop. Bij een tekstveld is het de waarde die zichtbaar is in het tekstveld op het scherm (en die over het algemeen is ingevuld door de gebruiker).

Het `placeholder`-attribuut wordt in de leereenheid over formvalidatie behandeld.

required-attribuut
Property

Het *required-attribuut* is een voorbeeld van een *property*. Zo'n attribuut krijgt geen waarde. De aan- of afwezigheid ervan bepaalt of het element een ingevulde *value* moet hebben of leeg gelaten mag worden.

Tekstveld

Door het `type`-attribuut van het `input`-element de waarde `text` te geven, krijgen we een *tekstveld* waarin slecht één regel tekst kan worden ingevoerd (we laten hier een tekstveld zien dat verplicht moet worden ingevuld).

```
<input type="text" required [overige-attributen] />
```

size-attribuut

Wanneer u wilt aangeven hoe breed het invoerveld is, kunt u het *size-attribuut* meegeven. Daarmee geeft u aan hoeveel karakters het invoerveld breed is. De waarde is het aantal karakters tussen aanhalingstekens.

label-element

Een goede omschrijving bij een control helpt een gebruiker enorm bij het invullen. Een *label-element* voegt deze omschrijving toe. Het `label`-element heeft een begin- en eindtag, en heeft tekst als inhoud. U kunt aangeven dat een `label`-element hoort bij een bepaald `input`-element door ze samen in een `div`-element of in een `span`-element te zetten (in de volgende leereenheid komen we terug op het verschil). Aan een control kunt u dus als volgt een label toevoegen:

```
<div>
  <label>labeltekst</label>
  <input type="text" name="woonplaats" />
</div>
```

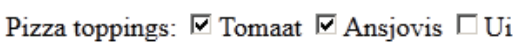
Met de tot nu toe behandelde elementen kunnen we eigenlijk alle soorten formulieren ontwikkelen, omdat de waarde van het `value`-attribuut altijd een string is. We bekijken nu nog twee `input`-elementen: de typen `checkbox` en `radio`, en de drop-downlijst. Door deze elementen te gebruiken, helpen we een gebruiker met het invullen van een formulier door het waardenbereik van een parameter sterk te verkleinen.

checkbox

Een *checkbox* wordt gebruikt als de items waaruit mag worden gekozen onafhankelijk van elkaar zijn en een gebruiker geen, één of meer items mag kiezen. Om een checkbox te creëren, gebruiken we het `input`-element waarbij het `type`-attribuut de waarde `checkbox` krijgt. Het volgende HTML-fragment toont een voorbeeld:

```
<div>
  <label>Pizza toppings:</label>
  <span>
    <input type="checkbox" name="tomaat" checked />
    <label>Tomaat</label>
  </span>
  <span>
    <input type="checkbox" name="ansjovis" checked />
    <label>Ansjovis</label>
  </span>
  <span>
    <input type="checkbox" name="ui" />
    <label>Ui</label>
  </span>
</div>
```

Figuur 2.2 toont de weergave in de browser, waarbij twee toppings zijn aangevinkt.



FIGUUR 2.2 Checkboxes

Wanneer een formulier met checkboxes wordt verstuurd, worden alleen de gegevens van de checkboxes die zijn aangevinkt naar de server gestuurd.

checked-attribuut

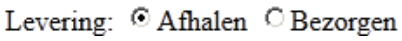
Bij checkboxes kan het *checked-attribuut* worden gebruikt. Dit attribuut is een property, en krijgt geen waarde. Andere voorbeelden van properties naast `required` zijn `disabled` en `readonly`.

radio button

Een *radio button* (keuzerondje) wordt gebruikt als een gebruiker slechts één item uit een aantal opties mag kiezen. Een radio button is een `input`-element waarbij het *type*-attribuut de waarde *radio* heeft. Het volgende HTML-fragment toont een voorbeeld:

```
<div>
  <span>
    <label>Levering:</label>
    <input type="radio"
           name="aflevering"
           value="afhalen"
           checked />
  </span>
  <span>
    <label>Afhalen</label>
    <input type="radio"
           name="aflevering"
           value="bezorgen" />
  </span>
  <label>Bezorgen</label>
</div>
```

Figuur 2.3 toont de weergave in de browser.



FIGUUR 2.3 Radio buttons

Let op!

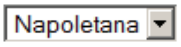
Zowel het `name`- als het `value`-attribuut zijn bij radio buttons van belang. Alle radio buttons met eenzelfde waarde van het `name`-attribuut vormen een groep. Binnen een groep kan maximaal één radio button geselecteerd zijn. Aan het `value`-attribuut kan het serverprogramma zien welke radio button is geselecteerd.

drop-downlijst

Een *drop-downlijst* maakt het mogelijk één of meerdere items in een lijst te selecteren. Het volgende HTML-fragment toont een voorbeeld:

```
<select name="pizza">
  <option value="napoletana">Napoletana</option>
  <option value="marinara">Marinara</option>
  <option value="viennese">Viennese</option>
</select>
```

Figuur 2.4 toont het bijbehorende element in de browser.



FIGUUR 2.4 Een drop-downlijst

select-element

size-attribuut

Een drop-downlijst wordt gevormd door een *select-element* met daarbinnen `option`-elementen die de keuzen binnen de drop-downlijst vormen. Het `select`-element kan een *size-attribuut* hebben dat aangeeft hoeveel items zichtbaar zijn in de lijst. Standaard heeft deze als waarde `1`.

option-element

selected-attribuut

De *option-element*en geven de keuzemogelijkheden aan. De inhoud van een `option`-element is in de browser te zien; de waarde van het `value`-attribuut wordt naar de server gestuurd wanneer de gebruiker de betreffende `option` kiest. U kunt het *selected-attribuut* meegeven om een bepaalde keuze standaard te selecteren. Het `selected`-attribuut is een property, en krijgt dus als het aanwezig is geen waarde.

multiple-attribuut

Wanneer een gebruiker meerdere items kan selecteren, moet het *multiple-attribuut* worden opgenomen. Ook dit attribuut is een property. Tevens moet één paar rechte haken (`[]`) aan het eind van de waarde van het `name`-attribuut worden geplaatst. Merk op dat als een gebruiker meerdere items mag selecteren, het beter is de waarde van het `size`-attribuut groter dan `1` te laten zijn.

optgroup-element
label-attribuut

Een ander zinvol element is het *optgroup-element*. Hiermee kunnen we een lijst opdelen in deellijsten. Een verplicht *label-attribuut* definieert de naam van elke sublijst. Hier volgt een voorbeeld van het gebruik van het `optgroup`-element. Let daarbij op het gebruik van het `size`- en `selected`-attribuut. Figuur 2.5 toont de weergave in een browser.

```
<select name="pizza[]" size="4" multiple>
  <optgroup label="Napolitaans">
    <option selected="" value="napoletana">
      Napoletana
    </option>
    <option value="marinara ">Marinara</option>
    <option value="margherita ">Margherita</option>
  </optgroup>
  <optgroup label="Lazio">
    <option value="viennese ">Viennese</option>
    <option value="capriciosa ">Capriciosa</option>
  </optgroup>
 </select>
```

FIGUUR 2.5 Een drop-downlijst met deellijsten

OPGAVE 2.8

Wanneer kunnen we het beste checkboxes, wanneer radio buttons en wanneer drop-downlijsten gebruiken?

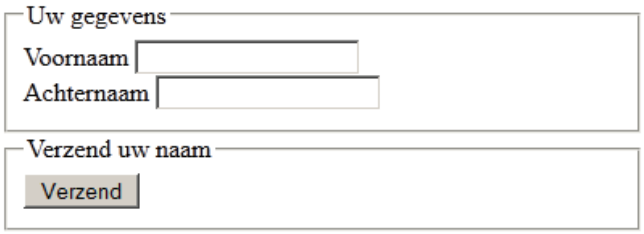
3.3 DE STRUCTUUR VAN FORMULIEREN

fieldset-element

Zoals u hebt gezien, groeperen `div`- en `span`-elementen zonder betekenis; het *fieldset-element* wordt gebruikt om meerdere controls betekenisvol te groeperen. Dit is zowel voor zoekmachines als voor mensen van belang. Met een `fieldset` groeperen we zowel in de HTML als op het scherm een aantal controls die logisch bij elkaar horen en voorzien we deze groep van een opschrift. De syntaxis is als volgt:

```
<fieldset>
  <legend>Opschrift</legend>
  <!-- userinterface elementen -->
</fieldset>
```

OPGAVE 2.9
Pas `hallo.html` zodanig aan dat u de volgende weergave in de browser krijgt:



FIGUUR 2.6 Hallo.html met fieldset- en div-elementen

In plaats van één `textarea`-element hebben we nu twee `input`-elementen van het type text met elk een eigen label.

3.4 STAPPENPLAN VOOR HET ONTWIKKELEN VAN EEN FORMULIER

Ten slotte presenteren we een algemeen stappenplan voor het ontwikkelen van een formulier. De aanpak is als volgt:

1 Bepaal welke gegevens de logica van de applicatie nodig heeft. In het geval van een client-serverprogramma is dat het server-programma. In het geval dat de applicatie alleen in de browser draait, is dat het JavaScript-programma.
2 Bepaal voor elk van deze gegevens het best bijpassende HTML-element en de daarbij horende attributen en attribuutwaarden. Denk daarbij expliciet aan duidelijkheid en gebruiksvriendelijkheid.
3 Maak een schets van het formulier en groepeer daarbij de gegevens op een voor de gebruiker logische manier. Voorzie elke groep van een bijschrift.
4 Implementeer de fieldsets.
5 Implementeer de label-control-combinaties. Gebruik hiervoor steeds een apart `div`-element.

In de zelftoets vragen we u aan de hand van dit stappenplan een formulier te ontwikkelen.

4 CSS, JavaScript en de webserver

De HTML-pagina is de basis. CSS voegt daar informatie aan toe over het uiterlijk; JavaScript voegt er logica of gedrag aan toe. Het principe van het scheiden van verantwoordelijkheden zegt dat informatie over uiterlijk, of logica en gedrag, niet thuishoren in de HTML.

4.1 HET ATTRIBUUT ID

id-attribuut

Aan elk element kan behalve het attibuut `class` en het attribuut `role` ook het *id-attribuut* worden meegegeven. Zoals de naam van dat attribuut al suggereert, moet zo'n `id` uniek zijn op de pagina.

Een voorbeeld van het gebruik daarvan is wanneer we bovenaan de pagina links willen opnemen om direct naar de verschillende secties te springen. Elke sectie krijgt dan een `id`, als volgt:

```
<h2 id="titletag">De title tag</h2>
<h2 id="metatag">De meta tag</h2>
```

In de introductie kunt u dan navigatie als volgt opnemen (bijvoorbeeld in de `td`-elementen van de tabel):

```
<td><a href="#titletag">title</a></td>
<td><a href="#metatag">meta</a></td>
```

Programmeeraanwijzing: Pas op met id
Wees uiterst terughoudend met het toekennen van een `id`. Kies in geval van twijfel voor een `class`. Dan kunt u zonder problemen een dergelijk element een tweede keer op de pagina gebruiken.

4.2 CSS

Wat CSS betreft is het enige dat in de HTML te zien is het feit dat er in het `head`-element een of meer CSS-bestanden bij de HTML horen (in `link`-elementen):

```
<link rel="stylesheet" href="stijl.css" />
```

Om in CSS te specificeren met welke HTML-elementen u precies iets wilt bereiken, kunt u onder andere de namen van tags, de `class`-attributen, de `id`-attributen en de `type`-attributen gebruiken. Die vormen als het ware de interface met CSS. Vaak is het nodig, wanneer u de pagina aan het opmaken bent met CSS, om meer elementen een `class`-attribuut of een `id`-attribuut mee te geven, maar wanneer u bij het opstellen van de HTML al op basis van de semantiek die attributen toevoegt wanneer dat zinvol lijkt, hebt u er over het algemeen weinig extra nodig.

4.3 JAVASCRIPT

Voor JavaScript geldt iets soortgelijks: u houdt het gescheiden van de HTML. U ziet alleen in de `head` dat er een of meer JavaScript-bestanden bij de pagina horen:

```
<script src="gedrag.js"></script>
```

De interface bestond vroeger alleen uit de namen van tags en de `id`-attributen, maar de interface wordt voortdurend uitgebreid. Wanneer u een JavaScript-library als JQuery gebruikt (binnen deze cursus zullen we dat doen) kunt u HTML-elementen binnen JavaScript op dezelfde manier als in het geval van CSS benaderen.

4.4 DE WEBSERVER

Wanneer een formulier wordt opgestuurd naar de webserver (naar de URL die genoemd wordt in het `action`-attribuut), krijgt de webserver de beschikking over de waarden van de `value`-attributen van de elementen van het formulier. De webserver kan die gegevens van elkaar onderscheiden doordat ze gelabeld zijn met de `name`-attributen. De `name`-attributen vormen dus de interface vanuit HTML naar de webserver.

ZELFTOETS

1 a Gegeven is de volgende omschrijving van het formulier voor het berekenen van de premie van een autoverzekering.
De premie is afhankelijk van de buurt waar iemand woont en zijn of haar leeftijd. De buurt waar iemand woont kan bepaald worden aan de hand van de postcode.
Van de auto moet het kenteken opgegeven worden, merk (bijvoorbeeld Volvo of Volkswagen), type (bijvoorbeeld V40, V50 en V70 voor het merk Volvo en Polo en Golf voor het merk Volkswagen), en de kleur.
Als het om een nieuwe auto gaat, of een auto die wordt geïmporteerd, moet het mogelijk zijn aan te geven dat het kenteken nog niet bekend is.
Gekozen kan worden uit een WA- of een All Risk-verzekering. Tevens kan de ingangsdatum aangegeven worden.
Bepaal welke gegevens de logica van de applicatie nodig heeft.
b Bepaal voor elk van deze gegevens of het al dan niet om een verplicht veld gaat.
c Bepaal voor elk van deze gegevens het best bijpassende HTML-element en de daarbij horende attributen en attribuutwaarden.
Denk daarbij expliciet aan duidelijkheid en gebruiksvriendelijkheid.
d Maak een schets van het formulier op papier en groepeer daarbij de gegevens op een voor de gebruiker logische manier.
Voorzie elke groep van een bijschrift.
e Implementeer de fieldsets.
f Implementeer de label-control-combinaties. Gebruik hiervoor steeds een apart `div`-element.

TERUGKOPPELING

1 Uitwerking van de opgaven

2.1 De validator geeft aan dat het document is geïnterpreteerd als HTML5: 'This document was successfully checked as HTML5!'. Er is geen melding van fouten (wel een waarschuwing over de afwezigheid van een character encoding).
De tweede keer geeft de validator aan dat er een fout in de HTML is gevonden: 'Line 4, Column 8: Element head is missing a required instance of child element title.' Het `title`-element is verplicht, dus wanneer het niet aanwezig is, verschijnt er een foutmelding. Ook hier is weer te zien dat de validator heeft herkend dat het om HTML5 gaat: 'Error found while checking this document as HTML5!'

2.2
```
<link rel="prev" href="tags.html" />
```

2.3 Omdat het gaat om een subsectie van een sectie met een `h2`-element voor de titel, gebruikt u voor de titel van de subsectie een `h3`-element:

```
<section>
  <h2>De meta tag</h2>
    ...
  <section>
    <h3>Namen van Metadata</h3>
    ...
  </section>
</section>
```

2.4 De code voor de tabs komt er nu als volgt uit te zien:

```
<nav class="tabs">
  <ul>
    <li class="tab active">HTML</li>
    <li class="tab">CSS</li>
    <li class="tab">JavaScript</li>
  </ul>
</nav>
```

2.5 De code is:

```
<p>De <strong>title-tag</strong> is verplicht aanwezig:
wanneer u een HTML-pagina zonder title-tag valideert, zult u
een foutmelding te zien krijgen.</p>
<p>De structuur van deze tag is simpel:
een <em>begintag</em>, een <em>eindtag</em>, en de titel
komt daartussen.</p>
```

2.6 a De code is:

```
<table>
  <thead>
    <tr>
      <th>Naam</th>
      <th>Beschrijving</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>author</td>
      <td>Auteur</td>
    </tr>
    <tr>
      <td>description</td>
      <td>Beschrijving</td>
    </tr>
    <tr>
      <td>keywords</td>
      <td>Steekwoorden</td>
    </tr>
  </tbody>
</table>
```

b Het element dat meer voor de hand ligt, is het `dl`-element: de description list. Dat geeft preciezer de betekenis weer van deze gegevens, en heeft dus de voorkeur. De tabel is te prefereren wanneer u het idee hebt dat u in de toekomst nog andere kolommen wilt toevoegen, zoals eisen aan de waarde van het `content`-attribuut voor alle namen.

2.7 De code is:

```
<p>Zoals u hier kunt zien, gebruiken zoekmachines het title-
element om een titel te geven aan zoekresultaten:
<img src="title-tags.png" width="400" height="222" alt="rood
omcirkelde titels van zoekresultaten" /></p>
```

en

```
<p>Zoals u hier kunt zien, gebruiken zoekmachines het meta
element met het attribuut name met als waarde description om
een beschrijving te geven bij de zoekresultaten:
<img src="meta-description.png" width="400" height="222"
alt="rood omcirkelde beschrijvingen van zoekresultaten"
/></p>
```

We hebben hier niet het `figure`-element gebruikt om het `img`-element mee te omhullen: we hebben het hier als een afbeelding binnen de tekst gebruikt. Inbedden in een `figure`-element is uiteraard mogelijk, en in feite semantisch gezien te prefereren; we willen hier laten zien dat een los `img`-element ook een mogelijkheid is.

2.8 Wanneer keuzes elkaar uitsluiten, zijn radio buttons of een lijst waarin slechts één item kan worden geselecteerd het handigst.
Wanneer het aantal items groot is, kan het beste gebruik worden gemaakt van een lijst.
Wanneer het aantal items klein is en meerdere items mogen worden geselecteerd, dan kan het beste van checkboxes gebruik worden gemaakt.
Overigens zijn deze grenzen niet scherp.

2.9 De gevraagde weergave kan verkregen worden door het toepassen van `fieldset`- en `div`-elementen.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hallo Service</title>
  </head>
  <body>
    <form method="get" action="server URL">
      <fieldset>
        <legend>Uw gegevens</legend>
        <div>
          <label>Voornaam</label>
          <input type="text" name="voornaam" value="" />
        </div>
        <div>
          <label>Achternaam</label>
          <input type="text" name="achternaam" value="" />
        </div>
      </fieldset>
      <fieldset>
        <legend>Verzend uw naam</legend>
        <input type="submit" value="Verzend" />
      </fieldset>
    </form>
  </body>
</html>
```

2 Uitwerking van de zelftoets

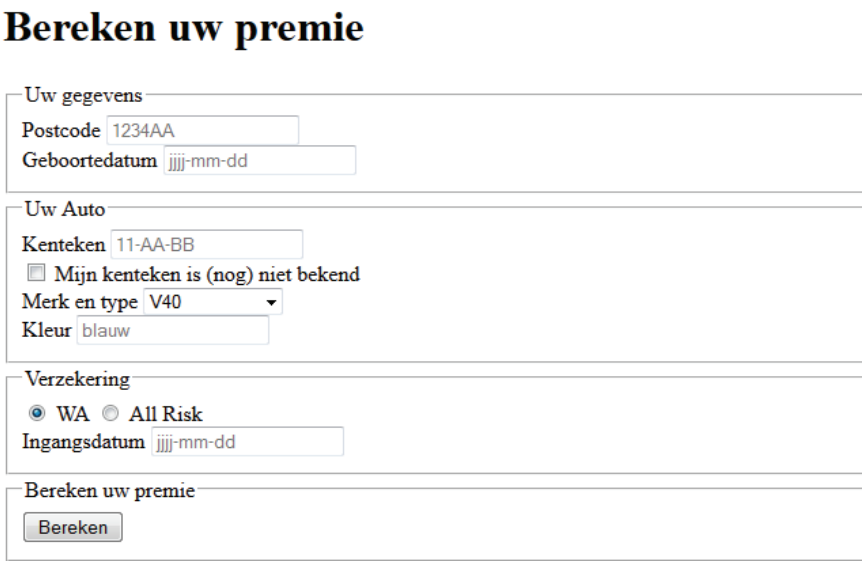
1 a Uit de casusomschrijving kunnen we de gegevens halen die nodig zijn:
– de leeftijd (we kiezen ervoor om de geboortedatum te laten invullen)
– de postcode
– het kenteken
– het merk auto (bijvoorbeeld Volvo)
– het type auto (bijvoorbeeld V40)
– de kleur van de auto
– het wel of niet bekend zijn van het kenteken
– het soort verzekering (WA of All Risk)
– de ingangsdatum van de verzekering.

b Verplicht zijn de volgende gegevens:
– de leeftijd
– de postcode
– het automerk
– het autotype
– de kleur
– het wel of niet bekend zijn van het kenteken
– het soort verzekering
– de ingangsdatum.

c Het best bijpassende HTML-element voor elk van deze gegevens is als volgt (we gebruiken hier ook het attribuut `placeholder`):
– Voor de geboortedatum kunnen we in principe een `input`-element van het type `text` nemen. Beter is een `input`-element van het type `date`. Het `name`-attribuut krijgt als waarde `geboortedatum`. Het `placeholder`-attribuut geven we waarde `jjjj-mm-dd`. Het element krijgt de property `required`.
– Voor de postcode kunnen we een `input`-element van het type `text` nemen. Het `name`-attribuut krijgt als waarde `postcode`. Het `placeholder`-attribuut geven we als waarde `1234AA`. Het element krijgt de property `required`.
– Voor het kenteken kunnen we ook een `input`-element van het type `text` nemen. Het `name`-attribuut krijgt als waarde `kenteken`. Het `placeholder`-attribuut geven we bijvoorbeeld de waarde `11-AA-BB`.
– Voor het automerk moet de gebuiker er één uit een (groot) aantal kiezen. Een drop-downlijst ligt dus voor de hand. Ditzelfde geldt voor het type auto. Daarbij geldt dat de autotypes worden bepaald door het automerk. Het ligt dus voor de hand hiervoor een drop-downlijst met deellijsten te nemen: per automerk worden de autotypes opgesomd. Het `name`-attribuut krijgt als waarde `merkEnType`. Vooraf selecteren we geen merk-type-combinatie. Het element krijgt de property `required`.
– Voor de kleur nemen we een `input`-element van het type `text` (type `color` is mooier, maar dat hebben we nog niet behandeld). Het `name`-attribuut krijgt de waarde kleur. Het `placeholder`-attribuut geven we als waarde `blauw`. Het element krijgt de property `required`.
– Voor het wel of niet bekend zijn van het kenteken gebruiken we een checkbox (default is `unchecked`). Het `name`-attribuut krijgt de waarde `kentekenBekend`.
– Voor het soort verzekering gebruiken we een `input`-element van het type `radio`. Het gaat om een keuze uit een korte lijst. Het `name`-attribuut krijgt als waarde `kenteken`. We kiezen ervoor om de keuze `WA` de property `checked` te geven.
– Voor de ingangsdatum kunnen we een `input`-element van het type `text` nemen (type `date` is mooier, maar we hebben dat nog niet behandeld). Het `name`-attribuut krijgt de waarde `ingangsdatum`. Het `placeholder`-attribuut geven we de waarde `jjjj-mm-dd`. Het element krijgt de property `required`.

d We kiezen ervoor om de volgende groepen van gegevens te onderscheiden:
– Uw gegevens, bestaande uit postcode en geboortedatum (deze zijn gebonden aan een persoon).
– Uw auto, bestaande uit het kenteken, het wel of niet bekend zijn van het kenteken, het merk en type en de kleur.
– Verzekering, bestaande uit de keuze WA of All Risk en de ingangsdatum.
– Bereken uw premie, die de submit-knop bevat.

Een schets van het formulier (in dit geval een weergave in de browser) ziet u in figuur 2.7



FIGUUR 2.7 Weergave van het formulier

e De eerste opzet met de `fieldset`-elementen is als volgt:

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Bereken direct de premie voor uw autoverzekering
    </title>
  </head>
  <body>
    <h1>Bereken uw premie</h1>
    <form method="get" action="server URL">
      <fieldset>
        <legend>Uw gegevens</legend>
        <!-- controls -->
      </fieldset>
      <fieldset>
        <legend>Uw Auto</legend>
        <!-- controls -->
      </fieldset>
      <fieldset>
        <legend>Verzekering</legend>
        <!-- controls -->
      </fieldset>
      <fieldset>
        <legend>Bereken uw premie</legend>
        <!-- controls -->
      </fieldset>
    </form>
  </body>
</html>
```

f De gehele implementatie is als volgt:

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Bereken direct de premie voor uw autoverzekering
    </title>
  </head>
  <body>
    <h1>Bereken uw premie</h1>
    <form method="get" action="server URL">
      <fieldset>
        <legend>Uw gegevens</legend>
        <div>
          <label>Postcode</label>
          <input type="text" name="postcode"
                 placeholder="1234AA" required />
        </div>
        <div>
          <label>Geboortedatum</label>
          <input type="date" name="geboortedatum"
                 placeholder="jjjj-mm-dd" required />
        </div>
      </fieldset>
      <fieldset>
        <legend>Uw Auto</legend>
        <div>
          <label>Kenteken</label>
          <input type="text" name="kenteken"
                 placeholder="11-AA-BB" />
        </div>
        <div>
          <input type="checkbox" name="kentekenBekend" />
          <label>Mijn kenteken is (nog) niet bekend</label>
        </div>
        <div>
          <label>Merk en type</label>
          <select name="merkEnType" required>
            <optgroup label="Volvo">
              <option value="VolvoV40">V40</option>
              <option value="VolvoV50">V50</option>
              <option value="VolvoV70">V70</option>
            </optgroup>
            <optgroup label="Volkswagen">
              <option value="VolkswagenPolo">Polo</option>
              <option value="VolkswagenGolf">Golf</option>
            </optgroup>
          </select>
       </div>
       <div>
          <label>Kleur</label>
          <input type="text" name="kleur"
                 placeholder="blauw" required />
       </div>
    </fieldset>
    <fieldset>
      <legend>Verzekering</legend>
```

```
      <div>
        <input type="radio" name="soortVerzekering"
               value="wa" checked />
        <label>WA</label>
        <input type="radio" name="soortVerzekering"
               value="allRisk" />
        <label>All Risk</label>
      </div>
      <div>
        <label>Ingangsdatum<label>
        <input type="text" name="ingangsdatum"
               placeholder="jjj-mm-dd" required />
      </div>
    </fieldset>
    <fieldset>
      <legend>Bereken uw premie</legend>
      <input type="submit" value="Bereken" />
      </fieldset>
    </form>
  </body>
</html>
```